УДК 004.4’24 https://doi.org/10.15407/pp2022.01.013

Я.О. Гайдукевич, А.Ю. Дорошенко

АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ ЗАПАСАМИ НА ОС ANDROID ТА БАЗИ ДАНИХ FIREBASE З ВИКОРИСТАННЯМ ШТРИХ КОДІВ ТА QR-КОДІВ

Розроблено новий програмний засіб для автоматизації запасами на основі ОС Android із використанням бази даних Firebase та мови програмування Java, що послуговується системою зчитування штрих-кодів та QR-кодів. Засіб забезпечує додавання товару на склад із подробицями про обраний товар та перегляд усіх запасів з їх ціною, категорією, назвою та кодом товару. Реалізовано авторизацію користувачів системи за допомогою відкритого стандарту FirebaseAuth та FirebaseDatabase. Створення єдиного сервісу для авторизації та реєстрації дозволило зробити систему масштабованою. У системі реалізовано пошук товару за кодом, а також перегляд запасів на складі з автоматичним підрахунком ціни. Основний сервіс бази даних – хмарна СУБД класу NoSQL, що дозволяє зберігати та синхронізувати дані між кількома клієнтами. Передбачено API для шифрування даних.

Ключові слова: ОС Android, Java, Firebase, авторизація, архітектура API, масштабовані системи автоматизації, NoSQL , СУБД , API для шифрування даних.

Вступ

Ми живемо в епоху завершення третьої цифрової революції, що почалася в другій половині минулого століття. Її характерні риси – розвиток інформаційно-комунікаційних технологій, автоматизація та роботизація виробничих процесів.

Розвиток інформаційних технологій і засобів комунікації, насамперед електронних мереж, створює потужний імпульс для формування нової тенденції функціонування сучасного бізнесу – діджиталізації економічних відносин. Більшість носіїв інформації стають цифровими, що визначає основний тренд розвитку як сучасної техніки, так і бізнес-процесів із переважною часткою електронної складової. Електронна форма комунікацій підвищує рівень і ефективність спілкування між покупцями та продавцями і створює нові ринки й можливості для реорганізації економічних процесів.

Автоматизована система управління запасами – найважливіший елемент діджиталізації підприємства. І з певного часу це вже не додаткова функція, а постійна потреба компаній, які хочуть конкурувати на ринку. Успішна діджиталізація передбачає комбінацію двох найважливіших аспектів: переведення всіх істотних активів у цифровий формат (наприклад, послуговування лише відсканованими документами, що зберігаються у хмарах) та використання спеціального програмного забезпечення для діджиталізації певних процесів компанії (наприклад, використання SAP для виконання різних транзакцій з постачальниками без урахування паперового документообігу). Цифрова трансформація - це серія загальних змін компанії, що приводить до підвищення ефективності та прибутковості компанії.

Якщо ви не знаєте, з чого почати автоматизацію вашого бізнесу, почніть із розробки індивідуальної системи управління запасами. Такий інструмент, створений безпосередньо під вашу компанію, завдяки автоматизації, допоможе м’яко перейти на електронний документообіг, водночас органічно вписуючись в існуючі процеси вашого підприємства. І пам’ятайте - завжди ухвалюйте рішення на основі даних. А ми допоможемо вам ці дані зібрати та обробити. Надалі такі системи інтегруватимуться з системами управління транспортними перевезеннями, системами автоматизації логістики запасів та закупівель. Метою даної статті є створення автоматизованої системи управління запасами на базі ОС Android для діджиталізації товарообігу підприємств.

Саме через товарообіг відбувається зміна форм вартості споживчих товарів, створеної в процесі виробництва. Традиційно товарообіг досліджується переважно на мікрорівні як основний показник діяльності торговельних підприємств. Однак на сьогодні дослідження товарообігу та факторів, які на нього впливають, має відбуватися і на макрорівні, оскільки оптимальна структура товарообігу держави є одним із головних чинників конкурентоспроможності її економіки.

Запаси – це активи підприємства (ресурси, майно), які:

• утримуються для подальшого продажу (розподілу, передачі) за умов звичайної господарської діяльності;

• перебувають у процесі виробництва з метою подальшого продажу продукту виробництва;

• утримуються для споживання під час виробництва продукції, виконання робіт та надання послуг, а також управління підприємством.

До запасів належать, зокрема:

• виготовлена на підприємстві готова продукція;

• сільськогосподарська продукція і продукція лісового господарства;

• придбані (отримані) товари, що утримуються підприємством з метою подальшого продажу;

• сировина, основні й допоміжні матеріали, комплектуючі вироби та інші матеріальні цінності, призначені для виробництва продукції, виконання робіт, надання послуг, розподілу, передачі, обслуговування виробництва й адміністративних потреб.

Збільшення запасів, з одного боку, приводить до підвищення ефективності, з іншого – зростає сума коштів, зокрема, на зберігання таких запасів. Водночас, скорочення запасів може призвести до збоїв процесів виробництва, поставок чи торгівлі.

Тому важливо мати оперативну та актуальну інформацію щодо кількості запасів підприємства на складі за категоріями, цінами тощо. Зі свого боку використання програмного забезпечення для управління запасами дає можливість усьому бізнесу працювати краще. Так, зокрема, вашим співробітникам більше не буде потрібно реєструвати кожну одиницю вручну, візуально перевіряти якість і кількість отриманого товару, адже такий підхід загрожує ризиком помилок людського фактора, а також вимагає великих тимчасових витрат.

Станом на даний момент існує декілька систем, які пропонують готові рішення автоматизованих систем управління виробництвом, зокрема, Replenishment+ від AbmCloud. Основне завдання системи – забезпечити постійну наявність сировини, матеріалів, комплектуючих у потрібному місці виробничого ланцюжка у потрібній кількості та у потрібний час. А також скоротити рівень надлишкових запасів, забезпечити високу надійність поставок та скоротити вплив зміни попиту на рівень запасів. Це дуже функціонально потужний конкурент, але для управління такою системою потрібні робочі місця та комп'ютери. Тоді як система, яку розробляє автор, націлена більше на діджиталізацію. Такою системою можна керувати з мобільного девайсу в будь-якому куточку світу, сканувати товари лише з телефону та додавати до інвентарю складу.

1. Формування вимог до системи

Система має давати можливість користувачу сканувати запаси за кодом товару. Можливість додавати обрані товари до інвентарю складу за ціною, назвою, категорією товару та штрих кодом. Для кожної позиції товару користувач може скористатися пошуком за штрих кодом, щоб вивести інформацію про виріб. Для кожного нового зареєстрованого акаунту в системі передбачена база даних, яка зберігає інформацію про користувача, а також про інвентар у його системі.

Система зі свого боку також повинна автоматично підраховувати кількість одиниць товару на складі та їхню цінову складову.

Як розшифровувати інформацію, закладену в QR-кодах? Це завдання успішно вирішує переважна частина сучасних смартфонів та планшетів. Для зчитування закодованих даних досить перейти у вікно додатку товару та активувати панель сканера, QR-код буде відсканований вбудованою камерою мобільного пристрою.

Замовники з постачальниками зможуть спілкуватися та обговорювати умови закупівлі нових запасів не боячись надлишків товару на складі, адже вся інформація буде вказана в додатку.

Діаграми використання системи для таких ролей користувача, як замовник, постачальник, та спостерігач (не авторизований користувач) представлено на Рисунку 1:

Розроблена діаграма використання відображає базовий набір функцій, які доступні авторизованому користувачу автоматизованої системи управління запасами на ОС Android, а саме мерченайдзеру (товарознавець чи помічник товарознавця, людина, яка представляє виробничу чи торговельну компанію у відповідних мережах. Відповідає за викладення товару, встановлення супутнього необхідного обладнання, розміщує POS-матеріали).

Оскільки система вбудована в мобільний девайс, то функціонал дає можливість реєструвати нові товари, використовуючи лише телефон або планшет, шукати вироби за номером штрих коду. Отже, у системі укомплектовано все необхідне в єдине ціле, в одну велику мережу, де можна зберігати та вести підрахунок товарів, не використовуючи стаціонарних комп'ютерів.

Система, орієнтована на потреби торговельної компанії в процесі автоматизації управління, надає такі можливості, як реклама нового товару, перегляд власних доданих товарів до інвентарю, пошук товару за штрих кодом, видалення товару, перегляд загальної кількості товарів та цін.

Після формування варіантів використання системи, було виділено мінімальні системні вимоги до виконання додатку. Перелік атрибутів наведено у Таблиці 1:

Таблиця 1. Мінімальні системні вимоги

Операційна система	Android 4.4 (KitKat) та вище
Оперативна пам'ять	1Гб та вище
Вбудована пам'ять	8Б та вище
GPS-модуль (A-GPS)	Так
Наявність сім-картки (GPRS)	рекомендуємо
Підтримка 3G (4G, LTE)	рекомендуємо
Фото-камера	5МП та вище
Процесор	Intel Atom® Processor Z2520 1.2 ГГц або швидший процесор
Пристрої зберігання даних	Між 850 Мбайтами та 1.2 ГБ
Жорсткий диск	2 ГБ доступного місця на жорсткому диску для встановлення; додатковий вільний простір потрібен під час встановлення.

Рис. 1. Діаграма використання мерченайдзером

Під час установки APK в системі Android встановлюється файл DEX, який містить код, ресурси тощо, скомпільовані як двійкові файли. Файл dex, зазвичай, має той самий розмір, що й файл apk, якщо у вас немає ресурсів, не скомпільованих в активах. Ще однією поширеною особливістю встановлення додатків є те, що android SAVES оригінальний apk при перевстановленні у разі помилок або з якихось інших причин також встановлюється. Ось чому обсяг пам'яті наших додатків у системі вдвічі більший. Це сума розміру вихідного apk та встановленого dex.

Проєктування бази даних починається в коді AndroidStudio, де визначаються типи даних для конкретного loyaute.xml файлу в Common Attributes. Структура проєктується в класах, конкретний клас програми може відповідати за окреме вікно, таке як, наприклад, реєстрація користувача, додавання продуктів, збереження їх у базі даних тощо. Підключення бази даних до проєкту також є невід'ємною частиною. До того ж для користування бази даних Firebase треба імпортувати відповідні бібліотеки:

```
import com.google.android.gms.tasks.
OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.
FirebaseAuth;
import com.google.firebase.auth.
FirebaseUser;
import com.google.firebase.database.
FirebaseDatabase;
```

Розроблена структура бази даних включає створення змінних із відповідними типами даних, а також розгалуження на дочірні елементи. Наступний фрагмент коду демонструє додавання елемента до бази даних:

```
public void additem(){
  String itemnameValue = itemname.
getText().toString();
  String itemcategoryValue = itemcategory.
getText().toString();
  String itempriceValue = itemprice.getText().
toString();
  String itembarcodeValue = itembarcode.
getText().toString();
   final FirebaseUser users = firebaseAuth.
getCurrentUser();
  String finaluser=users.getEmail();
   String resultemail = finaluser.
replace(".","");
  if (itembarcodeValue.isEmpty()) {
  itembarcode.setError("Порожньо ");
  itembarcode.requestFocus();
  return;
  }

  if(!TextUtils.isEmpty(itemnameValue)&&!
TextUtils.isEmpty(itemcategoryValue)&&!Te
xtUtils.isEmpty(itempriceValue)){

  Items items = new Items(itemnameValue,it
emcategoryValue,itempriceValue,itembarcod
eValue);
  databaseReference.child(resultemail).
child("Items").child(itembarcodeValue).
setValue(items);
  databaseReferencecat.
child(resultemail).child("ItemByCatego
ry").child(itemcategoryValue).
child(itembarcodeValue).setValue(items);
  itemname.setText("");
  itembarcode.setText("");
  itemprice.setText("");
  itembarcode.setText("");
  Toast.makeText(additemActivity.
this,itemnameValue+" Додано",Toast.
LENGTH_SHORT).show();
  }
  else {
  Toast.makeText(additemActivity.this,"Будь
ласка, заповніть усі поля",Toast.LENGTH_
SHORT).show();
  }
}
```

Таким чином розроблений фрагмент коду додавання товарів до бази даних.

2. Реалізація бізнес-логіки

Бізнес-логіка – це частина коду, яка виконує логіку додатку. Крім бізнес-логіки, у додатках може бути код, який відповідає за відображення інформації, управління інформацією, роботу із зовнішніми ресурсами/сервісами.

У системі бізнес-логіка зосереджена в класах, які синхронізовані з базою даних, що зберігає інформацію у форматі обміну

даних JSON. Крім цього при запитах на модифікацію даних передаються відповідні заголовки авторизації для уникнення несанкціонованого доступу.

Нижче наведено декілька реалізацій класів.

Клас scanItemsActivity призначений для роботи зі штрих кодами та QR кодами, наприклад, пошук товару за номером коду. Клас утворює два вікна: одне – пошук товару за кодом, а інше – вивід товару з параметрами, такими як: назва виробу, його код, ціна і категорія.

Також викликається збережена процедура, яка збирає із бази даних інформацію та повертає результат у форматі JSON. Клас сервісу складається з наступних методів:

1. метод firebasesearch. Метод створює запит до бази даних, щодо номера коду товару, який має повернути або вивести на екран, а також створює об'єкт FirebaseRecyclerAdapter, який зв'язується з Query до RecyclerView. Коли дані додаються, видаляються або змінюються, ці оновлення автоматично застосовуються до інтерфейсу користувача в реальному часі;

```
        public void
firebasesearch(String searchtext){
        Query firebaseSearchQuery
= mdatabaseReference.
orderByChild("itembarcode").
startAt(searchtext).
endAt(searchtext+"\uf8ff");

FirebaseRecyclerAdapter<Items,
        UsersViewHolder>
firebaseRecyclerAdapter = new
FirebaseRecyclerAdapter<Items,
UsersViewHolder>
         ( Items.class,
         R.layout.list_layout,
         UsersViewHolder.class,
         firebaseSearchQuery )
         {
```

2. метод setDetails. Метод створює TextView для виводу їх на екран додатку після пошуку конкретного товару, а саме код товару, назву, категорію та ціну.

За аналогічною схемою реалізовано клас (viewInventoryActivity) для роботи з інвентарем складу, але виводяться всі товари без пошуку.

```
        public void setDetails(Context
ctx,String itembarcode, String
itemcategory, String itemname, String
itemprice){
        TextView item_barcode =
(TextView) mView.findViewById(R.
id.viewitembarcode);
        TextView item_name =
(TextView) mView.findViewById(R.
id.viewitemname);
        TextView item_category =
(TextView) mView.findViewById(R.
id.viewitemcategory);
        TextView item_price =
(TextView) mView.findViewById(R.
id.viewitemprice);

        item_barcode.
setText(itembarcode);
        item_category.
setText(itemcategory);
        item_name.setText(itemname);
        item_price.setText(itemprice);
         }
```

3. Метод onCreate. Метод працює з базою даних, створює точку входу для доступу до бази даних Firebase. Можливість отримати екземпляр, використавши getInstance(). Щоб отримати доступ до розташування в базі даних та прочитати або записати дані, створено getReference().

```
        @Override
         protected void
onCreate(Bundle savedInstanceState)
{
         super.
onCreate(savedInstanceState);
         setContentView(R.layout.
activity_scan_items);
         firebaseAuth = FirebaseAuth.
getInstance();
         final FirebaseUser users =
firebaseAuth.getCurrentUser();
```

```
        String finaluser=users.
getEmail();
        String resultemail = finaluser.
replace(“.”,””);
        mdatabaseReference =
FirebaseDatabase.getInstance().
getReference(“Users”).
child(resultemail).child(“Items”);
        resultsearcheview =
findViewById(R.id.searchfield);
        scantosearch
= findViewById(R.
id.imageButtonsearch);
        searchbtn = findViewById(R.
id.searchbtnn);

        mrecyclerview =
findViewById(R.id.recyclerViews);
        LinearLayoutManager
manager = new
LinearLayoutManager(this);
        mrecyclerview.
setLayoutManager(manager);
        mrecyclerview.
setHasFixedSize(true);

        mrecyclerview.
setLayoutManager(new
LinearLayoutManager(this));
        }
```

Клас RegisterActivity призначений для роботи із користувачами системи управління запасами, реєстрації їхніх акаунтів та збереження у базі даних, щоб зберігати тільки ту інформацію про товар, яку додавав конкретний користувач. Конфіденційність даних у базі присутня, тому паролі не виводяться задля безпеки даних. Клас складається з наступних методів:

1. метод onCreate. Містить логіку реєстрації користувача, створює текстові поля для заповнення даних, таких як емейл, пароль, повтор пароля та ім’я . Щоб отримати об’єкт FirebaseAuth, викликає статичний метод getInstance().

```
    @Override
        protected void
onCreate(Bundle savedInstanceState)
{
        super.
onCreate(savedInstanceState);
        setContentView(R.layout.
activity_register);

        editTextName =
findViewById(R.id.departmentName);
        editTextEmail =
findViewById(R.id.emailRegister);
        editTextPassword
= findViewById(R.
id.passwordRegister);
        editTextcPassword=
findViewById(R.id.confirmPassword);
        UserRegisterBtn=
findViewById(R.id.button_register);
        progressBar =
findViewById(R.id.progressbar);
        progressBar.
setVisibility(View.GONE);
        mAuth = FirebaseAuth.
getInstance();

        UserRegisterBtn.
setOnClickListener(new View.
OnClickListener() {
        @Override
        public void onClick(View v) {
        registerUser();
        }
        });
        }
```

2. метод onStart. Обробляє вже зареєстрованого користувача.
3. метод registerUser. Обробляє поля, що були у методі onCreate, надає їм тип даних .toString(), Якщо поля неправильно заповнені або пусті видає відповідне повідомлення. Метод отримує дані та надсилає дані на основі електронної пошти користувача до бази даних, а також виводить повідомлення у разі успішної або неуспішної реєстрації.

Клас additemActivity потрібен для додавання товару до інвентарю складу. Також він створює поля для вводу даних про вибір, таких як: назва товару, категорія, ціна та числовий тип даних для вводу номера коду товару. Крім того, клас від-

слідковує зміну в полях під час зберігання товару та додає до бази даних. Клас складається з наступних методів:

1. метод onCreate. Містить логіку реєстрації товару, створює текстові поля для заповнення даних, таких як: назва товару, категорія, ціна та числовий тип даних для вводу номера коду товару. Щоб отримати об'єкт FirebaseAuth, викликається статичний метод getInstance(). Нижче наведено реалізацію:

```
@Override
  protected void
onCreate(Bundle savedInstanceState)
{
    super.
onCreate(savedInstanceState);
    setContentView(R.layout.
activity_additem);
    firebaseAuth = FirebaseAuth.
getInstance();

    databaseReference =
FirebaseDatabase.getInstance().
getReference("Users");
    databaseReferencecat =
FirebaseDatabase.getInstance().
getReference("Users");
    resulttextview =
findViewById(R.id.barcodeview);
    additemtodatabase
= findViewById(R.
id.additembuttontodatabase);
    scanbutton =
findViewById(R.id.buttonscan);
    itemname = findViewById(R.
id.edititemname);
    itemcategory=
findViewById(R.id.editcategory);
    itemprice = findViewById(R.
id.editprice);
    itembarcode=
findViewById(R.id.barcodeview);


    scanbutton.
setOnClickListener(new View.
OnClickListener() {
    @Override
    public void onClick(View
view) {
      startActivity(new
Intent(getApplicationContext(),
ScanCodeActivity.class));
      }
      });

      additemtodatabase.
setOnClickListener(new View.
OnClickListener() {
      @Override
      public void onClick(View v) {
      additem();
      }
      });
    }
```

2. метод additem. Виконує такі функції, як: додавання елемента до бази даних, створення полів для їх заповнення, передавання цих полів до бази даних, зберігання.
3. метод Logout. Потрібен для виклику меню у верхній частині екрану та виходу з акаунту. Цей метод фігурує також у інших класах, де є можливість виклику меню. Наведено реалізацію нижче:

```
    private void Logout()
{
 firebaseAuth.signOut();
 finish();
 startActivity(new
Intent(additemActivity.
this,LoginActivity.class));
  Toast.makeText(additemActivity.
this,"LOGOUT SUCCESSFUL",
Toast.LENGTH_SHORT).show();
}
```

4. Метод onCreateOptionsMenu. Потрібен для MenuInflater - це системний ресурс Android. Створюється під час завантаження андроїда. Це постійний об'єкт, і посилання на нього завжди доступне у пам'яті. Кожен підклас класу Context, тобто Activity може отримати посилання на нього, викликавши getMenuInflater()зсередини класу. Наведено нижче реалізацію:

```
        @Override
public boolean
onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.
menu,menu);
  return true;
}
```

Клас ScanCodeActivity. Відповідає за сканування штрих кодів та QR кодів. Нижче наведено реалізацію (використано бібліотеку ZXingScannerView scannerView):

```
        import com.google.zxing.
Result;
        import me.dm7.
barcodescanner.zxing.
ZXingScannerView;

        public class ScanCodeActivity
extends AppCompatActivity
implements ZXingScannerView.
ResultHandler {
         int MY_PERMISSIONS_
REQUEST_CAMERA=0;

         ZXingScannerView
scannerView;
         @Override
         protected void
onCreate(Bundle savedInstanceState)
{
         super.
onCreate(savedInstanceState);
         scannerView = new
ZXingScannerView(this);

setContentView(scannerView);
         }

         @Override
         public void
handleResult(Result result) {

         additemActivity.
resulttextview.setText(result.
getText());

         onBackPressed();
         }

         @Override
         protected void onPause() {
         super.onPause();
         scannerView.stopCamera();
         }

         @Override
         protected void
onPostResume() {
         super.onPostResume();
         if (ContextCompat.checkSelf
Permission(getApplicationContext(),
Manifest.permission.CAMERA)
         != PackageManager.
PERMISSION_GRANTED)
         {
         ActivityCompat.
requestPermissions(this, new String[]
{Manifest.permission.CAMERA},
         MY_PERMISSIONS_
REQUEST_CAMERA);
         }
         scannerView.
setResultHandler(this);
         scannerView.startCamera();
}}
```

Таким чином було реалізовано панель сканування для штрих кодів та QR кодів.

3. Тестування

У структурі Android Studio є папки androidTest і test поруч з основною папкою з класами проєкту.

Навіщо потрібне тестування? Якщо додаток маленький, то тести не потрібні й цілком можна обходитися без тестування і далі. Чому? Йдеться про те, що в невеликих проєктах можна контролювати логіку програми. Також можна передбачити слабкі місця та виправити код.

Але все змінюється, якщо програма стала складною. Якщо з'явилося понад десятка різних екранів активностей, окремих класів тощо, код слід розбивати на модулі, аби забезпечити незалежність. Такий підхід обов'язково використовується у компаніях, де кожен відповідає за свою ділянку коду.

Тести поділяються на дві категорії – локальні (Unit Testing) та інструментальні (UI Testing).

Локальні тести перевіряють роботу методу, класу, компонента. Тест не залежить від Android. Ппрактично, перевіряється код Java, який можна контролювати на звичайному комп'ютері без участі пристрою або емулятора. Наприклад, такому варіанту відповідає додавання двох чисел типу int. Подібні тести проводять у папці Test.

Для інструментальних тестів наявність пристрою або емулятора є обов'язковою, оскільки потрібно тестувати натискання кнопки, введення тексту, прокручування, торкання та інші операції. Тести проводять у папці androidTest.

Висновки

Реалізовано програмний засіб для автоматизації запасами на основі ОС Android із використанням бази даних Firebase та мови програмування Java. Система використовує метод зчитування штрих кодів та QR-кодів. Він забезпечує додавання товару на склад з усіма подробицями про обраний товар та перегляд усіх запасів на складі з їхньою ціною, категорією, назвою та кодом товару. Реалізовано авторизацію користувачів системи за допомогою відкритого стандарту FirebaseAuth та FirebaseDatabase.

Надано можливість реєструвати користувачів системи, що дозволило зробити систему масштабованою. У системі реалізовано пошук товару за кодом, а також перегляд запасів на складі з автоматичним підрахунком ціни. Основний сервіс бази даних - хмарна СУБД класу NoSQL, що дозволяє зберігати та синхронізувати дані між кількома клієнтами. Передбачено API для шифрування даних. Система у грошовому еквіваленті автоматично розраховує результати управління запасами товарів та загальну ціну.

Бібліотека me.dm7.barcodescanner.zxing.ZXingScannerView використовувалась для реалізації сканера.

Запропонована автоматизована система управління запасами уможливлює обрання найбільш ефективних інструментів для регулювання товарообігу та управління запасами.

References

1. Stateless 3.0 - A State Machine library for .NET Core [Online] - Access mode: https://www.hanselman.com/blog/stateless- 30-a-state-machine-library-for-net-core /.
2. Problems of information technology software development for supercomputer systems [Online]. - Access mode https://zakon.rada.gov.ua/rada/show/v0347550-10#Text.
3. Architectural Styles and the Design of Network-based Software Architectures [Online] - Access mode: https://www.ics.uci.edu/~fielding/pubs/dissertation / top.htm
4. Alexander Wald, Paul Datel, Harvey Datel. Android for developers. 3rd edition, 2016
5. John Horton. Learning Java by Building Android Games - Explore Java Through Mobile Game Development, 2019.
6. Greg Nudelman. Android Design Patterns: Interaction Design Solutions for Developers, 2013.
7. Mark L. Murphy. Busy Coder's Guide to Android Development, 2020.
8. John Darwin. Android. Collection of recipes, 2018.
9. Don Griffiths, David Griffiths. Head First. Programming for Android, 2016.
10. Robert Cecil Martin. Net code - 2008.
11. Christine Marsicano, K. Stewart, Bill PhillipsAndroid. Programming for Professionals, 4th Edition, 2021.

Література

1. Stateless 3.0 – бібліотека State Machine для .NET Core [Online] – Режим доступу: https://www.hanselman.com/blog/stateless-30-a-state-machine-library-for-net-core/ .
2. Проблеми розробки програмного забезпечення інформаційних технологій для суперкомп'ютерних систем [Online] – Режим доступу:https://zakon.rada.gov.ua/rada/show/v0347550-10#Text.
3. Архітектурні стилі та дизайн архітектур програмного забезпечення на основі мережі [Online] – Режим доступу: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.
4. Олександр Уолд, Пол Датель, Харві Датель. Android для розробників. 3-е видання, 2016.
5. Джон Хортон. Вивчення Java шляхом створення ігор для Android – Досліджуйте Java за допомогою розробки мобільних ігор, 2019.

6. Грег Нудельман. Шаблони дизайну Android: рішення для дизайну взаємодії для розробників, 2013.
7. Марк Л. Мерфі. Посібник Busy Coder з розробки Android, 2020.
8. Джон Дарвін. Android. Збірник рецептів, 2018.
9. Дон Гріффітс, Девід Гріффітс. Голова спочатку. Програмування для андроїд, 2016.
10. Роберт Сесіл Мартін. Чистий код – 2008 р.
11. Крістін Марсікано, К. Стюарт, Білл ФілліпсAndroid. Програмування для професіоналів, 4-е видання, 2021.

Отримано 18.12.2021

***Про авторів*:**

Гайдукевич Ярослав Олегович,
магістрант кафедри інформаційних систем та технологій КПІ імені Ігоря Сікорського. Кількість наукових публікацій в українських виданнях – 1. http://orcid.org/0000-0002-6300-1778,

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук, професор, завідувач відділу теорії комп'ютерних обчислень Інституту програмних систем НАН України, професор кафедри автоматики і управління в технічних системах НТУУ «КПІ імені Ігоря Сікорського». Кількість наукових публікацій в українських виданнях – понад 180. Кількість наукових публікацій в іноземних виданнях – понад 70.
Індекс Гірша – 6.
http://orcid.org/0000-0002-8435-1451

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.
e-mail: doroshenkoanatoliy2@gmail.com, yarmcfly@gmail.com